Komplexitätstheorie

WiSe 2011/12 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Ein Beispiel Lineare Suche nach Wert `x` im Array `A[0,n-1]`:

```
i = 0;
while ((i < n) && (A[i] != x))
   i++;
```

In welchem Sinne wird hier nach `x` gesucht ?

Was passiert, wenn `x` gefunden wird ?

Was passiert, wenn `x` nicht gefunden wird ?

Arbeitet das Programm(fragment) stets “wie beabsichtigt” ?

Ein Beispiel Lineare Suche nach Wert `x` im Array `A[0,n-1]`:

```
i = 0;
while ((i < n) && (A[i] != x))
   i++;
```

Wieviele “Schritte” führt der Algorithmus im schlimmsten Falle aus ?
Eingangs erfolgt eine Zuweisung.
Jedesmal, wenn die Schleife betreten werden soll, werden zwei Tests durchgeführt und die (Booleschen) Ergebnisse per Konjunktion verknüpft.
Mit dem nachfolgenden Inkrement (im Schleifenrumpf) werden “im negativen Fall” (`x` wird nicht gefunden) je Schleifeniteration vier “Operationen” durchgeführt.
Im schlimmsten Fall wird `x` nicht gefunden, es werden also `n` (vergebliche) Schleifendurchläufe durchgeführt sowie noch ein abschließender negativer Test der ersten Bedingung `(i < n)`.
Insgesamt sind das also `4n+2` Operationen.
Aufgabe: Haben wir wirklich alles “richtig gezählt” ?
Welcher Annahmen haben wir bei dieser Art von Zählung (implizit) gemacht ?

Was waren unsere impliziten Annahmen bei der Analyse ?

Wir hatten fünf unterschiedliche Operationen “einfach gleich” gezählt:
`i = 0;` ist eine *Zuweisung* einer Konstanten zu einer Variablen.
`i < n` ist ein *Vergleich* zweier Variablenwerte.
`A[i] != x` ist ebenfalle ein Vergleich; er beinhaltet aber darüber hinaus den *Zugriff auf ein Element in einem Feld*.
`i++` ist ein *Inkrement* (Hochzählen) einer Variablen.

Je nach konkreter Maschine und je nach konkretem Compiler sind die genannten Operationen “tatsächlich” unterschiedlich schnell und auch untereinander verschieden.

Registermaschinen / RAMs

Maschinenmodell:

* Speicher potentiell unendlicher Größe, realisiert als Register
* Register können zunächst ganze, “manchmal” sogar reelle Zahlen aufnehmen
* Befehlssatz analog Assemblersprachen:

- Lade-/Speicherbefehle,
- arithmetische, logische etc. Operationen,
- Sprungbefehle

* beschreibt nicht: Parallelrechner, Speicherhierarchie, ...

Das RAM-Modell Eigenschaften

Die Maschine hat einen (zentralen) Prozessor.
Eine RAM arbeitet sequentiell (keine Parallelverarbeitung von Befehlen).
Jeder Schritt der Maschine verursacht Kosten 1, unabhängig von der Größe der Operanden (*Einheitskostenmaß*); nur manchmal werden wir davon abweichen...
Zu den “Schritten” der Maschine zählen auch direkte und indirekte Speicherzugriffe.
Indirekte Speicherzugriffe gestatten die effiziente “RAM-Implementierung” von Feldern.

Registermaschinen im Detail

Wie schon das letzte Mal ausgeführt, finden Sie zu dieser Vorlesung sehr viel sehr gutes Begleitmaterial im Internet.

Exemplarisch führe ich den Foliensatz von Wolfgang Schreiner aus Linz im Folgenden vor.
Früher: `http://moodle.risc.uni-linz.ac.at/file.php/2/03-ram.pdf`
Lokale Kopie: RAM-Linz.pdf

Beobachte: RAMs sind sehr hardwarenah, gleichzeitig aber noch recht nah an elementarem Pseudo-Code.
$\leadsto$ Wir können RAM-Programme stets (bequemer) in Pseudo-Code notieren.